SoftSide® Selections

A product of **SoftSide** Publications, Inc.
10 Northern Blvd., Amherst, NH 03031

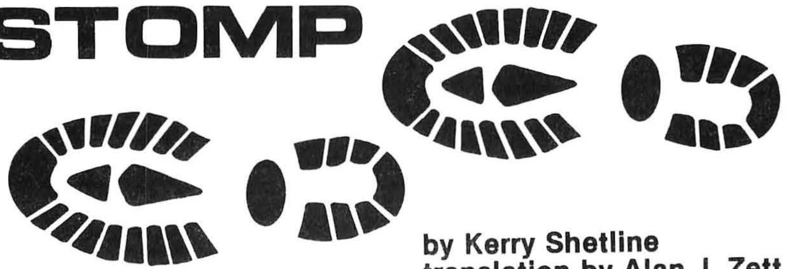
SoftSide Selections

ATARI® is a registered trademark of Warner Communications.

Index

This booklet prepared by the staff of SoftSide Publications, Inc. Entire contents copyright © 1983 by SoftSide Publications, Inc., 10 Northern Blvd., Northwood Executive Park, Amherst, NH 03031

STOMP



by Kerry Shetline
translation by Alan J. Zett

***STOMP* is a utility to help track down typing errors in programs from *SoftSide Selections.* It replaces *SWAT* beginning with Issue 45.**

Note: Please read *all* the instructions for *STOMP* before beginning.

Why *STOMP?*

The purpose of *STOMP (Stop Typos On My Programs)* is to aid the reader who types BASIC programs from a listing in ***SoftSide*** or ***SoftSide Selections.*** It is ***SoftSide's*** new official debugging utility. Until now, this function was filled by *SWAT* which readers hailed as a great aid in tracking down the inevitable errors that occur when you type several pages of BASIC code. The concise *SWAT* table narrows the search for errors to no more than twelve lines of code. For complex programs, you can reduce this number at will.

Nonetheless, *SWAT* has drawbacks and deficiencies. For one thing, to get a matching *SWAT* table, readers must type every line of the program exactly as it appears in the magazine. Even REM statements, which BASIC ignores, have to be replicated precisely. Also, *SWAT* can't detect simple transpositions. This means that the numbers 32767 and 36277 are identical to *SWAT.* If these numbers are part of a DATA statement for a machine language routine, the computer may hang up, or important data may vanish. Furthermore, because *SWAT* is written entirely in BASIC, it is quite slow. In sum, *SWAT,* although a big help, leaves much room for improvement.

STOMP corrects *SWAT's* deficiencies. It is faster, easier to use, and more reliable than *SWAT.* In addition, *STOMP* ignores REM statements and insignificant spaces. If you type BASIC programs from ***SoftSide Selections,*** *STOMP* will save you many hours.

You may omit any REM statements that appear in our programs. If you do so, be sure to remove any colon (:) immediately preceding the word REM. In addition, feel free to add REM statements (within the constraints of memory) without changing the *STOMP* tables.

The tables *STOMP* generates are identical in appearance to *SWAT* tables, so *SWAT* veterans will have no trouble interpreting *STOMP* tables. Type *STOMP* in now, and start using it right away!

Starting Up With *STOMP*

STOMP is a hybrid of machine language and BASIC. The machine language portion of *STOMP* occupies a string in the BASIC program. To create *STOMP,* type the listing for the *STOMP Generator* below. Save this program, check it with the procedure below, and run it.

Checking *STOMP Generator* And Saving *STOMP*

The first thing the program asks you is whether to save *STOMP* to disk or cassette. To test the program, however, press "E" to run the program without actually saving *STOMP* (the material that would go to the magnetic media goes instead to the screen). If everything is proper, the program lines that contain the machine language appear on the screen, and are entered into the program. Then, the message "Generating STOMP..." is displayed, and the *STOMP* program appears.

If you see the message, "Error in DATA lines 100-140...," re-load the *Generator* program, and carefully check those lines for errors.

Now, in lines 32190 and 32230, change the number 125 to 128. List these lines, then use your cursor movement keys to move the cursor over the number 5 in each line, type over it with the "8" key, and press Return.

Next, type GOTO 32000. This creates a *STOMP* table for the *STOMP Generator*. Compare this table to the published one. If they match, then re-load the *STOMP Generator,* and run it. This time, select "D" or "C" to save the *STOMP* program to disk or cassette.

If you choose to save *STOMP* on cassette, the Atari buzzes twice. This signals you to prepare a cassette and press the Return key. If you choose disk, the *STOMP Generator* automatically saves *STOMP* under the file name "STOMP" on drive 1.

Using *STOMP*

First, type and save the program you are going to check. Then follow these simple steps:

1. Place the disk or cassette on which you saved *STOMP* in the disk drive or tape recorder.
2. Add *STOMP* to your program with the ENTER command.
   Disk: ENTER "D:STOMP"
   Tape: ENTER "C:"
3. Press the Return key once for disk, or twice for tape.
4. Type GOTO 32000
5. Press the Return key.
6. When *STOMP* asks you for the name of the program, press Return, or type an identifying name.
7. If the published *STOMP* table lists modified "parameters," type them in response to the next prompt, line count first, then byte count, separated by a comma. For example, if the table says: "Modified Parameters: Line count = 3, Byte count = 200," you would type "3,200" and press Return in response to this prompt. If no special parameters are listed, then simply press Return.

If you have a printer turned on and connected to your Atari, *STOMP* automatically sends its output there. With or without a printer, the *STOMP* table appears on the screen. Compare the table you get with the one we publish along with the program you are checking. If they match, you may go ahead and enjoy the program. If not, see the section below.

What To Do If The *STOMP* Tables Don't Match

● First examine the listed line numbers in the first columns of the tables. If they don't match, it probably means you have inserted, omitted or changed at least one line number. A large omission from a line can also throw off the first column. An extra or missing line affects all entries in the first column from that point on. Search the lines indicated by the first erroneous entry for the bad line, and correct it. Then try running *STOMP* again to see if the entries match.

● If, after you have verified all the line numbers, there are still discrepancies in the second or third column, you need to make a more detailed search.

(a) If the length for a range of lines is too big, then it is likely that you typed too many spaces or other characters in a PRINT or DATA statement. Check all variable names.

(b) If the length entry is too small, check for omissions. Did you leave out part of a line? Did you miss a character in a PRINT or DATA statement?

(c) If the length is correct but the *STOMP* code is wrong, check the appropriate lines for transposed or mistyped characters.

Remember that *STOMP* ignores REM statements and spaces that aren't significant. Anything within quotation marks is vital, and you must type it precisely. The same goes for DATA statements, except for spaces between the word DATA and the first item, which are not important.

What's All This About Parameters?

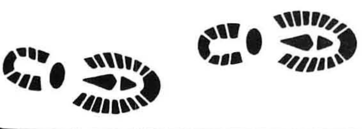
Parameters are simple to understand. Normally, *STOMP* starts a new line in its table when it has checked twelve lines or 500 bytes of memory, whichever comes first. This means it will narrow your search for typographical errors to about a twelve-line range of the program. If a program is particularly complex, we make *STOMP* generate new table entries at smaller intervals, making your search for typos still easier.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS         Atari BASIC         SS
SS      'STOMP Generator'      SS
SS   Author: Kerry Shetline    SS
SS  Translator: Alan J. Zett   SS
SS      Copyright © 1983       SS
SS SoftSide Publications, Inc  SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

If you don't wish to type this program, it is available on Issue #45 SoftSide DV and CV.

Initialize and choose an output device for the STOMP program.

```
10 GRAPHICS 0:CLR :DIM A$(1),F$(7):? "
OUTPUT TO DISK OR CASS ";:INPUT A$:IF
A$="" OR A$<"C" OR A$>"E" THEN 10
```



Change the device letter in A$ into a device specification in F$: "C:" for cassette or "D:STOMP" for disk. For initial program checking, the device specification "E:STOMP" may also be generated.

```
20 F$=A$:F$(2)=":":IF A$="C" THEN 40
30 F$(3)="STOMP"
```

Calculate the checksum of the machine language DATA, and display a message if the sum is incorrect.

```
40 Z=0:FOR X=1 TO 152:READ Y:Z=Z+Y:NEX
T X:IF Z<>18401 THEN GRAPHICS 0:? "ERR
OR IN DATA LINES 100-140...":END
```

Erase the screen, read the first half of the DATA, and create the first dummy line of BASIC.

```
50 GRAPHICS 0:RESTORE :POKE 766,1:? :?
 :? "32050 S$=";CHR$(34);:FOR X=1 TO 7
6:READ Y:? CHR$(Y);:NEXT X:?
```

Read the second half of the DATA, and create the second dummy line of BASIC. Also set up the CONT for line 70.

```
60 ? "32060 S$(77)=";CHR$(34);:FOR X=7
7 TO 152:READ Y:? CHR$(Y);:NEXT X:POKE
 766,0:? :? "CONT"
```

Point the cursor to fall on the first dummy line, POKE the Atari into the continuous input mode, and break out of the program to enter the new lines of BASIC. The Atari will enter the two lines, reach the CONTinue statement, and execute the rest of the program, starting at line 80.

```
70 POSITION 2,0:POKE 842,13:STOP
```

Clear the continuous input mode, and generate STOMP by LISTing the STOMP lines of STOMP Generator to the device specified in F$.

```
80 POKE 842,12:GRAPHICS 0:? "GENERATIN
G STOMP...":LIST F$,32000,32767:END
```

Data for the machine language portion of STOMP to be contained in S$.

```
100 DATA 104,240,21,104,133,204,104,13
3,203,169,0,133,205,133,206,133,207,14
1,0,6,141,1,6,96,160,0,162,0,142,2,6
110 DATA 200,177,203,208,251,200,232,1
89,3,6,209,203,208,15,201,45,208,243,1
73,0,6,133,212,173,1,6,133,213,96,160
120 DATA 0,177,203,208,14,173,2,6,74,1
76,8,200,208,243,230,205,24,144,225,15
2,72,173,2,6,208,21,162,0,177,203,221
130 DATA 3,6,208,12,201,45,208,4,104,2
4,144,227,200,232,208,237,104,168,177,
203,201,2,208,10,72,173,2,6,73,1,141
140 DATA 2,6,104,132,212,6,212,69,212,
24,109,0,6,141,0,6,144,3,238,1,6,230,2
06,208,180,230,207,208,176
```

Set up page-6 pointers, and test whether a printer is ready and on-line. The variable P will be set to 0 if the printer is not ready, and 1 if it is.

```
32000 POKE 1539,26:POKE 1540,50:POKE 1
541,37:POKE 1542,45:CLR :TRAP 32010:CL
OSE #1:OPEN #1,8,0,"P":P=1
32010 TRAP 33333:GRAPHICS 0:POSITION 6
,1:? "ATARI STOMP BY ALAN J. ZETT":? :
? :? :? "PROGRAM NAME ";:DIM A$(20)
```

Print title, prompt for program name, dimension strings, input the program name and new STOMP parameters (if not default).

```
32020 DIM S$(152),Z$(8):Z$="        ":
INPUT A$:TRAP 32030:? :? "STOMP PARAME
TERS ";:INPUT NU,B:GOTO 32040
32030 NU=12:B=500
```

Set up two graphics 0 screens; one to display the STOMP table, and the other to hold each line of BASIC code as it is listed into screen memory. In this way, the lines can be scanned in ASCII by the machine code portion and remain invisible to the user. The ASCII method eliminates SWAT's problem with the variable table, so listing programs to cassette or disk is now unneccesary.

```
32040 TRAP 33333:GRAPHICS 0:T=PEEK(106
):L=T-4:S=L-4:POKE 106,L:GRAPHICS 0
```

Lines 32050 and 32060 contain the machine language routine in the variable S$. These lines will be created by STOMP Generator.

Find the memory location of the first line of BASIC and check whether a program name has been specified.

```
32070 POSITION 2,23:NL=PEEK(136)+PEEK(
137)*256:IF A$="" THEN 32090
```

Print the program name and modified parameters, if specified.

```
32080 ? "ATARI STOMP TABLE FOR ";A$:?
:IF P THEN ? #1;"ATARI STOMP TABLE FOR
 ";A$:? #1
32090 IF NU=12 AND B=500 THEN 32110
32100 ? "PARAMETERS ARE: ";NU;" AND ";
B:? :IF P THEN ? #1;"PARAMETERS ARE: "
;NU;" AND ";B:? #1
```

Print header information for STOMP tables.

```
32110 POKE 752,1:? "     LINES       C
ODE   LEN":? "-----------------------
---"
32120 IF P THEN ? #1;"     LINES
 CODE   LEN":? #1;"-------------------
--------"
```

Initialize S$ to look at the second screen area, and reset important pointers between line ranges. Also reset the first-line pointer FL.

```
32130 X=USR(ADR(S$),L*256+104):FL=-1
```

Calculate this-line (TL) and next-line (NL) pointers, the current line number (LN), and the more significant byte of the next line number (Q).

```
32140 TL=NL:NL=TL+PEEK(TL+2):LN=PEEK(T
L)+PEEK(TL+1)*256:Q=PEEK(NL+1)
```

List one line of BASIC code to the alternate screen, and execute the STOMP machine language subroutine. Check whether the line just listed is a valid first line number for the range (REMs do not count as a valid line number; they are ignored).

```
32150 POKE 106,T:POKE 82,0:POKE 89,L:?
 CHR$(125);:LIST LN:CS=USR(ADR(S$)):IF
 PEEK(205)=1 AND FL=-1 THEN FL=LN
```

Point back to the first screen and store the number of bytes in the last line STOMPed into the variable BC.

```
32160 POKE 82,2:POKE 89,S:POKE 106,L:B
C=PEEK(206)+PEEK(207)*256
```

Check whether the maximum number of lines in a line range has been counted by STOMP. (Line count = 12 for default parameters.)

```
32170 IF PEEK(205)=NU THEN 32200
```

Check whether the maximum number of bytes in a line range has been counted by STOMP. (Byte count = 500 for a default parameter.)

```
32180 IF BC>=B THEN 32200
```

If the more significant byte of the next line number is less than the start of the STOMP program, then continue listing codes in the table.

```
32190 IF Q<125 THEN 32140
```

Calculate a STOMP code from the line range checksum and display one line of the STOMP table.

```
32200 POSITION 2,23:CS=CS-INT(CS/676)*
676:C1=INT(CS/26):C2=CS-C1*26+65:? Z$(
1,6-LEN(STR$(FL)));FL;" - ";LN;
32210 ? Z$(1,9-LEN(STR$(LN)));CHR$(C1+
65);CHR$(C2);Z$(1,7-LEN(STR$(BC)));BC:
IF P=0 THEN 32230
32220 ? #1;Z$(1,6-LEN(STR$(FL)));FL;"
- ";LN;Z$(1,9-LEN(STR$(LN)));CHR$(C1+6
5);CHR$(C2);Z$(1,7-LEN(STR$(BC)));BC
```

Set the checksum and the quote flag to zero, and continue listing codes in the table if the last program line before STOMP hasn't been processed yet.

```
32230 POKE 205,0:POKE 206,0:POKE 207,0
:IF Q<125 THEN 32130
```

Reset important pointers and end the program.

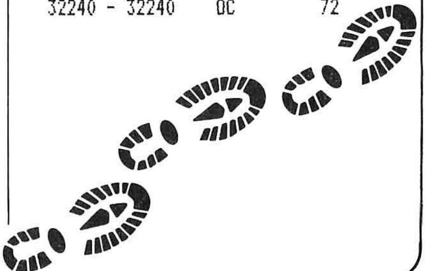
```
32240 POKE 106,T:POKE 89,L:? CHR$(125)
:POKE 89,S:POSITION 2,23:POKE 752,0:?
:END
```

STOMP TABLE

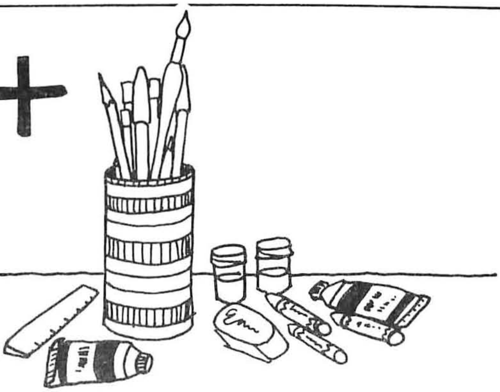
For **ATARI® STOMP GENERATOR**

(Modified Parameters:
Line count = 3, byte count = 200)

LINES	STOMP CODE	LENGTH
10 - 30	DW	143
40 - 60	PM	271
70 - 100	PN	207
110 - 120	UZ	222
130 - 140	YB	210
32000 - 32020	TW	291
32030 - 32050	YT	170
32060 - 32080	ML	229
32090 - 32110	FL	191
32120 - 32140	CH	181
32150 - 32170	RP	185
32180 - 32200	LC	146
32210 - 32230	MH	249
32240 - 32240	OC	72



Draw 7+



by Steven Chanin

Draw 7+ is a graphics editing program for an Atari® with 40K RAM, one disk drive and a joystick.

The Atari personal computer's most striking feature is its graphics. The high-resolution multicolored pictures featured in many new games are amazing. Unfortunately, for the average user, the graphics modes used by the professionals are not standard. Previously, the only way to create these visual masterpieces was with expensive graphics editors, or by doing an in-depth study of machine language, modified Display Lists and many other difficult topics — until now. *Draw 7+* lets your artistic urges flow for free, using what I call GRAPHICS 7+, an ANTIC mode not implemented in BASIC. GRAPHICS 7+ is a half-height GRAPHICS 7 screen, so it requires two separate screens to build the display. With its 160 horizontal by 192 vertical resolution in four colors, you can create pictures rivaling those of many commercial programs.

Most of *Draw 7+* is in BASIC, so it is short, powerful and easy to modify. Later in this article, I will take the program apart piece by piece so you can see how to use its techniques in your own programs. Because *Draw 7+* contains machine language routines, you should save your copy before running it.

For those who just want to draw, read the instructions below and run the program. *Draw 7+* has three modes: Draw, Expand and Color.

Draw Mode

Use the Draw Mode to create your picture. Pushing the joystick in any of its eight directions moves the cursor in that direction. Pressing the fire button plots the point beneath the cursor. Pressing the fire button while pushing the stick draws a line in the desired direction.

Draw Commands

- To change the cursor's drawing color, enter a number from zero to three. Colors 1, 2 and 3 are foreground colors. Color 0 is the background color and drawing in it is equivalent to erasing.
- SELECT saves a picture to disk under the current file name. If another picture with the same name has been saved, the new picture will replace it. To change the file name, see the Name Mode.
- OPTION loads the picture from the disk under the current file name. To change the file name, see the Name Mode.

● S clears the GRAPHICS 7+ screen, resets the color values and returns the cursor to the center of the screen. The program will ask if you want to erase the screen. Type N or Y and press return.
● C sends you to the Color Mode.
● N sends you to the Name Mode. It lets you change the name of the current picture. Pressing N prints the current file name and a prompt, which asks if you want to change it. Type N or Y and press return. When typing the new name be sure to prefix it with "D:". Note: Use an extension of .PIC for clarity.
● E expands the area around the cursor from GRAPHICS 7+ to GRAPHICS 3 so you can do detailed work. See the Expand Mode instructions.
● H displays a menu of all the *Draw 7+* commands (for all three modes).
● L draws a flashing "cross hair" to help you locate the cursor (if you can't find the flashing dot). To resume drawing, press any key.

Expand Mode

The Expand Mode is similar to the Draw Mode. It also features movement and drawing in eight directions and four colors.

● D compresses the GRAPHICS 3 screen into the box on the GRAPHICS 7+ screen and returns you to the Draw Mode.

Color Mode

In the Color Mode you may change the hues and intensities for colors 1 through 3, as well as for the background. The changes are not permanent (i.e., they are not saved with the picture) so you can modify the colors without fear of "ruining" your picture. When you arrive in the Color Mode you are modifying color 1. Color changes are retained until you type either the command "S" or stop the program. A saved picture which had modified colors will have "standard" colors when you reload it from disk.
● To change the cursor's drawing color, enter a number from zero to three. See Draw commands.
● SELECT increases the hue of the color you are modifying. For example, if you were working on color 1 (pink) and you hit SELECT twice, color 1 becomes blue. Changing the hue has no effect on the intensity.
● OPTION increases the intensity of the color you are modifying. For example, if you were working on color 1 and it had an intensity of 5, hitting OPTION would change the intensity to 6. Note: You will have to press OPTION twice for it to affect the screen, because only even intensities are valid.
● D returns you to the Draw Mode (modified hues and intensities are retained).

Logical Operators

One of the most useful but ignored BASIC operations is the logical comparison. In Atari BASIC the statement "A>B" will return a one if A is greater than B, or a zero if A is smaller than B or they are equal. This technique can, if used correctly, replace a lengthy series of "IF...THEN" statements. As an example of this technique I will explain a line from the *Draw 7+* program listing. Here is line 970:

```
970  XP=XP+1:XB=XB+(XP=4):XP=XP-4*(XP=4):XB=XB-
(XB=40):RETURN
```

The following IF...THEN statements replace line 970:

```
          XP=XP+1
          IF XP=4 THEN XB=XB+1
          IF XP=4 THEN XP=0
          IF XB=40 THEN XB=39
          RETURN
```

Line 970 begins by incrementing XP. If XP=4 then the expression (XP=4) equals 1 and SB is incremented. If XP=4 then 4*(XP=4) equals 4 and XP is reduced to zero. IF XB=40 then (XB=40) equals 1 and XB is reduced to 39, otherwise (XB=40) equals 0 and SB stays the same. Finally the line returns to the line which called it.

One of this program's most powerful features is the graphic expansion routine. It displays, in GRAPHICS 3, a portion of the GRAPHICS 7+ screen around the cursor. It sets up a GRAPHICS 3 screen below the GRAPHICS 7+ screen and copies the screen memory data around the cursor to the GRAPHICS 3 screen.

Having two co-resident screens is easier than it looks. Just lower the high memory pointer (106 or $6A) so the top of memory starts below the higher Display List and screen memory. After you push the pointer down, a normal BASIC GRAPHICS command (i.e. GRAPHICS 3) sets up the new screen and takes care of all the system pointers for you. To return to the first screen, restore the high memory pointer to its original value, and execute another GRAPHICS command with the clear screen option disabled (accomplished by adding 32 to the value specified in the GRAPHICS command). The only problem with this technique is that the values in the color registers are reset to the default values; therefore, you must save and restore them after the GRAPHICS command is finished. Lines 1210 and 1220 do this.

Because so much has been written about modifying Display Lists, I will only give a brief synopsis of the technique. I set the value of the variable DL with the Display List Pointer in RAM. This insures that the program will run on computers with different amounts of memory. Stepping through the Display List, I changed the mode line bytes from fifteen (GRAPHICS 8) to fourteen (GRAPHICS 7+). Finally, I changed the LMS bytes from a 79 (GRAPHICS 8 w/LMS option) to a 78 (GRAPHICS 7+ w/LMS option). This Display List modification was simple because both GRAPHICS 7+ and GRAPHICS 8 have 192 vertical mode lines and require the same amount of memory.

Feel free to modify the program to your heart's delight. If you renumber the program after making modifications, be sure to check the branches from line 140 to lines 150-170 because these do *not* work with increments of ten. Exercise the same care with lines 730-760 because these also will not work with increments of ten. I will be glad to answer questions about this program — just write to me c/o ***SoftSide.***

Variables

A$: Holds reply to yes and no questions.
B: Memory location of GRAPHICS 7+ cursor in memory.
C: Current drawing color (in Expand and Draw Modes).
C1-C4: Temporary storage of screen colors.
CONSOL: Value of Atari console keys (START, SELECT, OPTION, AND SYSTEM RESET).
CORNER: Memory location of the upper left hand corner of the GRAPHICS 7+ cursor window.

DL: Location of Display List in memory.
E: Offset from current cursor position to the end of the GRAPHICS 7+ cursor window (used by the expansion and locate routines).
FN$: Filename used for disk I/O.
KEY: Value of last key pressed.
L: Offset from the current cursor position to the left side of the GRAPHICS 7+ cursor window (expansion routine).
R: Offset from the current cursor position to the right side of the GRAPHICS 7+ cursor window (expansion routine).
RAMTOP: Top of memory pointer.
S$: Contains the stored screen data (locate routine).
SIZHI,SIZLO: Variables used for disk I/O.
T: Offset from the current cursor position to the top of the GRAPHICS 7+ cursor window (expansion and locate routines).
TVAHI,TVALO,TVSIZ: Variables for disk I/O.
XB: Horizontal location of the byte containing the GRAPHICS 7+ cursor.
XLEN: Width of the GRAPHICS 7+ cursor window.
XP: Horizontal position of a selected pixel in XB.
YB: Vertical location of the byte containing the GRAPHICS 7+ cursor.
YLEN: Length of the GRAPHICS 7+ cursor window.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                            SS
SS         Atari BASIC        SS
SS          'Draw 7+'         SS
SS   Author:  Steven Chanin   SS
SS     Copyright © 1983       SS
SS SoftSide Publications, Inc SS
SS                            SS
SS SS SS SS SS SS SS SS SS SS SS
```

If you don't wish to type this program, it is available on Issue #45 SoftSide DV and CV.

Intialization.

```
20 GRAPHICS 24:DIM FN$(17),A$(1),S$(5)
:POKE 77,0:DL=PEEK(560)+256*PEEK(561):
GOSUB 1010
```

Set up machine language routines (see also *Listings 1 and 2).

```
30 FOR X=1536 TO 1722:READ Y:POKE X,Y:
NEXT X
40 DATA 104,104,104,141,255,6,104,104,
141,254,6,104,104,141,253,6,173,255,6,
24,106,106,106,141,255
50 DATA 6,169,63,172,253,6,240,7,74,74
,9,192,136,208,249,141,252,6,173,254,6
,45,252,6,141
60 DATA 254,6,173,255,6,172,253,6,240,
5,74,74,136,208,251,141,252,6,173,254,
6,13,252,6,133
70 DATA 212,169,0,133,213,96
80 DATA 104,104,133,91,104,133,90,104,
104,141,255,6,104,104,141,254,6,104,10
4,141,253,6,160,0,173
90 DATA 253,6,208,7,177,90,145,88,24,1
44,4,177,88,145,90,200,204,255,6,208,2
34,165,90,24,105
100 DATA 40,133,90,165,91,105,0,133,91
,165,88,24,105,10,133,88,165,89,105,0,
133,89,206,254,6
110 DATA 173,254,6,201,0,208,196,96
120 DATA 104,201,1,208,10,104,104,141,
114,3,162,48,32,86,228,133,213,169,0,1
33,212,96,0
```

More initialization.

```
130 XB=19:XP=0:YB=96:C=1:FN$="D:PICTUR
E1.PIC":POKE 764,255
```

Process joystick movements.

```
140 CLOSE #3:KEY=PEEK(764):CONSOL=PEEK
(53279):POKE 77,0:TRAP 140:GOTO 140+2*
STICK(0)
150 GOSUB 970:GOSUB 990:GOTO 240
152 GOSUB 970:GOSUB 1000:GOTO 240
154 GOSUB 970:GOTO 240
158 GOSUB 980:GOSUB 990:GOTO 240
160 GOSUB 980:GOSUB 1000:GOTO 240
162 GOSUB 980:GOTO 240
166 GOSUB 990:GOTO 240
168 GOSUB 1000:GOTO 240
170 GOTO 240
```

Draw Mode commands.

```
240 B=DL+40*YB+XB+282:IF STRIG(0)=0 TH
EN X=USR(1536,C,PEEK(B),XP):POKE B,X:G
OTO 140
250 IF CONSOL<7 OR KEY<255 THEN 260
252 X=PEEK(B):Y=USR(1536,AZ,X,XP):POKE
 B,Y:AZ=AZ+1:IF AZ=4 THEN AZ=0
254 FOR Z=1 TO 10:NEXT Z:POKE B,X:GOTO
 140
260 IF KEY=31 THEN C=1:POKE 764,255:GO
TO 140
270 IF KEY=30 THEN C=2:POKE 764,255:GO
TO 140
280 IF KEY=26 THEN C=3:POKE 764,255:GO
TO 140
290 IF KEY=50 THEN C=0:POKE 764,255:GO
TO 140
300 IF CONSOL=5 THEN X=11:GOTO 390
310 IF CONSOL=3 THEN X=7:GOTO 390
320 IF KEY=18 THEN X=708:Y=INT(PEEK(X)
/16):Z=PEEK(X)-16*Y:POKE 764,255:GOTO
420
330 IF KEY=35 THEN POKE 764,255:GOTO 5
50
340 IF KEY=62 THEN POKE 764,255:GOTO 6
10
350 IF KEY=42 THEN POKE 764,255:GOTO 6
50
360 IF KEY=57 THEN POKE 764,255:GOTO 1
030
370 IF KEY=0 THEN POKE 764,255:GOTO 11
40
380 POKE 764,255:GOTO 140
```

Disk Save/Load routine.

```
390 OPEN #3,X-3,0,FN$:RAMTOP=PEEK(106)
*256:TVSIZ=RAMTOP-DL
400 SIZHI=INT(TVSIZ/256):SIZLO=TVSIZ-S
IZHI*256:TVAHI=INT(DL/256):TVALO=DL-TV
AHI*256
410 POKE 884,TVALO:POKE 885,TVAHI:POKE
 888,SIZLO:POKE 889,SIZHI:Z=USR(1691,X
):CLOSE #3:GOTO 140
```

Color Mode.

```
420 CONSOL=PEEK(53279):KEY=PEEK(764)
430 IF KEY=31 THEN X=708:POKE 764,255:
GOTO 540
440 IF KEY=30 THEN X=709:POKE 764,255:
GOTO 540
450 IF KEY=26 THEN X=710:POKE 764,255:
GOTO 540
460 IF KEY=50 THEN X=712:POKE 764,255:
GOTO 540
470 IF CONSOL=5 THEN Y=Y+1:Y=Y-16*(Y=1
6):GOTO 520
480 IF CONSOL=3 THEN Z=Z+1:Z=Z-16*(Z=1
6):GOTO 520
490 POKE X,16*Y+Z
500 IF KEY=58 THEN POKE 764,255:GOTO 1
40
510 GOTO 420
520 IF PEEK(53279)<>7 THEN 520
530 GOTO 490
540 Y=INT(PEEK(X)/16):Z=PEEK(X)-16*Y:G
OTO 420
```

Change Name.

```
550 POKE 106,PEEK(106)-32:GOSUB 1210:G
RAPHICS 0:POSITION 7,7:PRINT "OLD NAME
 =";FN$
560 POSITION 10,10:PRINT "CHANGE IT (Y
/N)";:INPUT A$:IF A$<>"Y" AND A$<>"N"
THEN 560
570 IF A$="N" THEN POKE 106,PEEK(106)+
32:GRAPHICS 56:GOSUB 1220:GOSUB 1010:G
OTO 140
580 POSITION 10,13:PRINT "NAME (WITH D
:)":POSITION 10,14:INPUT FN$
585 IF LEN(FN$)<3 THEN PRINT CHR$(253)
;CHR$(253);CHR$(253):GOTO 580
590 IF FN$(1,2)<>"D:" THEN PRINT CHR$(
253);CHR$(253);CHR$(253):GOTO 580
600 POKE 106,PEEK(106)+32:GRAPHICS 56:
GOSUB 1220:GOSUB 1010:GOTO 140
```

Erase screen command.

```
610 POKE 106,PEEK(106)-32:GOSUB 1210:G
RAPHICS 0:POSITION 13,9:PRINT "ERASE S
CREEN":POSITION 13,11
620 PRINT "SURE (Y/N)";:INPUT A$:IF A$
<>"Y" AND A$<>"N" THEN 620
630 POKE 106,PEEK(106)+32:IF A$="N" TH
EN GRAPHICS 56:GOSUB 1220:GOSUB 1010:G
OTO 140
640 GRAPHICS 24:GOSUB 1010:XP=0:XB=19:
YB=96:GOTO 140
```

Expand GRAPHICS 7+ window to GRAPHICS 3 screen.

```
650 POKE 106,PEEK(106)-32:R=XB+4:L=XB-
5:T=YB-12:E=YB+11
```

```
660 IF R>39 THEN R=R-1:GOTO 660
670 IF L<0 THEN L=L+1:GOTO 670
680 IF T<0 THEN T=T+1:GOTO 680
690 IF E>191 THEN E=E-1:GOTO 690
700 XLEN=R-L+1:YLEN=E-T+1:CORNER=PEEK(
88)+256*PEEK(89)+40*T+L
710 GOSUB 1210:GRAPHICS 19:COLOR C:D1=
PEEK(88):D2=PEEK(89):D3=PEEK(90):D4=PE
EK(91)
720 X=USR(1617,CORNER,XLEN,YLEN,0):POK
E 88,D1:POKE 89,D2:POKE 90,D3:POKE 91,
D4:X=20:Y=12
```

Process joystick movements.

```
730 TRAP 730:KEY=PEEK(764):GOTO 730+2*
STICK(0)
740 GOSUB 930:GOSUB 950:GOTO 830
742 GOSUB 930:GOSUB 960:GOTO 830
744 GOSUB 930:GOTO 830
748 GOSUB 940:GOSUB 950:GOTO 830
750 GOSUB 940:GOSUB 960:GOTO 830
752 GOSUB 940:GOTO 830
756 GOSUB 950:GOTO 830
758 GOSUB 960:GOTO 830
760 GOTO 830
```

Expand Mode commands.

```
830 IF STRIG(0)=0 THEN PLOT X,Y:FOR Z=
1 TO 30:NEXT Z:GOTO 730
840 IF KEY=31 THEN C=1:POKE 764,255:GO
TO 730
850 IF KEY=30 THEN C=2:POKE 764,255:GO
TO 730
860 IF KEY=26 THEN C=3:POKE 764,255:GO
TO 730
870 IF KEY=50 THEN C=0:POKE 764,255:GO
TO 730
880 IF KEY=58 THEN POKE 764,255:GOTO 9
00
890 LOCATE X,Y,Z:COLOR AZ:PLOT X,Y:FOR
 W=1 TO 15:NEXT W
892 COLOR Z:PLOT X,Y:FOR W=1 TO 15:NEX
T W:AZ=AZ+1:IF AZ=4 THEN AZ=0
894 COLOR C:GOTO 730
900 D1=PEEK(88):D2=PEEK(89):D3=PEEK(90
):D4=PEEK(91):X=USR(1617,CORNER,XLEN,Y
LEN,1)
910 POKE 88,D1:POKE 89,D2:POKE 90,D3:P
OKE 91,D4
920 POKE 106,PEEK(106)+32:POKE 764,255
:GOSUB 1210:GRAPHICS 56:GOSUB 1010:GOT
O 140
```

Change cursor position (Expand Mode).

```
930 X=X+(X<39):RETURN
940 X=X-(X>0):RETURN
950 Y=Y+(Y<23):RETURN
960 Y=Y-(Y>0):RETURN
```

Change cursor position (Draw Mode).

```
970 XP=XP+1:XB=XB+(XP=4):XP=XP-4*(XP=4
):XB=XB-(XB=40):RETURN
980 XP=XP-1:XB=XB-(XP=-1):XB=XB+(XB=-1
):XP=XP+4*(XP=-1):RETURN
990 YB=YB+1:YB=YB-(YB=192):RETURN
1000 YB=YB-1:YB=YB+(YB=-1):RETURN
```

Set up GRAPHICS 7+.

```
1010 FOR X=DL+6 TO DL+99:POKE X,14:NEX
T X:FOR X=DL+102 TO DL+198:POKE X,14:N
EXT X
1020 POKE DL+3,78:POKE DL+99,78:RETURN
```

Help Menu.

```
1030 POKE 106,PEEK(106)-32:GRAPHICS 0:
POSITION 14,0:PRINT "DRAW MODE":PRINT
"BUTTON - PLOT POINT"
1040 PRINT "STICK  - MOVE CURSOR":PRIN
T "# (0-3)- CHANGE COLOR":PRINT "SELEC
T - SAVE"
1050 PRINT "OPTION - LOAD":PRINT "S
   - CLEAR SCREEN":PRINT "C      - COL
OR MODE"
1060 PRINT "N      - CHANGE NAME":PRIN
T "E      - EXPAND SCREEN":PRINT "H
   - HELP (THIS SCREEN)"
1070 PRINT "L      - LOCATE CURSOR":PO
SITION 13,13:PRINT "COLOR MODE":PRINT
"# (0-3)- CHANGE COLOR"
1080 PRINT "SELECT - INCREASE HUE":PRI
NT "OPTION - INCREASE INTENSITY":PRINT
 "D      - DRAW MODE"
1090 POSITION 16,19:PRINT "EXPAND":PRI
NT "STICK  - MOVE   ";CHR$(25);"D
 - DRAW MODE"
1100 PRINT "BUTTON - PLOT   ";CHR$(25)
;"# (0-3)- CHANGE COLOR";
1110 PRINT :POSITION 8,23:PRINT "HIT A
NY KEY TO RETURN";
1120 IF PEEK(764)=255 THEN 1120
```

```
1130 POKE 106,PEEK(106)+32:POKE 764,25
5:GRAPHICS 56:GOSUB 1010:GOTO 140
```

Locate command.

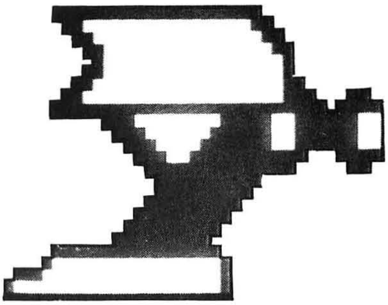
```
1140 T=B-80:E=B+80:X=USR(1536,C,0,XP)
1150 IF T<DL+282 THEN T=T+40:GOTO 1150
1160 IF E>DL+7961 THEN E=E-40:GOTO 116
0
1170 FOR Y=T TO E STEP 40:S$((Y-T)/40+
1)=CHR$(PEEK(Y)):POKE Y,X:NEXT Y:POKE
B,85*C
1180 FOR Y=T TO E STEP 40:POKE Y,ASC(S
$((Y-T)/40+1)):NEXT Y
1190 IF PEEK(764)=255 THEN 1170
1200 POKE 764,255:GOTO 140
```

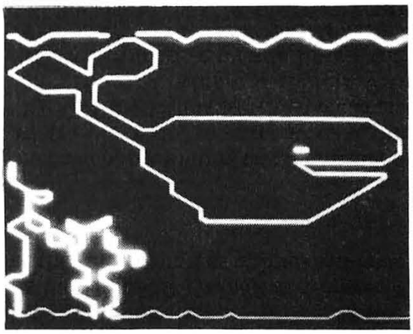
Store color registers (in C1-C4).

```
1210 C1=PEEK(708):C2=PEEK(709):C3=PEEK
(710):C4=PEEK(712):RETURN
```

Return color values to color registers.

```
1220 POKE 708,C1:POKE 709,C2:POKE 710,
C3:POKE 712,C4:RETURN
```





STOMP TABLE

For **ATARI® DRAW 7 +**

(Modified Parameters:
Line count = 5, byte count = 300)

LINES	STOMP CODE	LENGTH
20 - 50	TW	301
60 - 100	OG	389
110 - 150	BV	281
152 - 162	IM	124
166 - 250	OV	155
252 - 280	AY	212
290 - 330	QC	202
340 - 380	VB	159
390 - 430	ZL	301
440 - 480	BK	208
490 - 530	XN	95
540 - 580	ID	330
585 - 620	IV	334
630 - 670	FG	226
680 - 720	EZ	273
730 - 748	LF	146
750 - 760	SP	95
830 - 870	JJ	203
880 - 900	LX	236
910 - 950	AO	175
960 - 1000	LT	203
1010 - 1050	ST	357
1060 - 1090	LP	368
1100 - 1140	XV	241
1150 - 1190	ZY	218
1200 - 1220	EG	139

DV BONUS

PIX

by Peter J. Favaro

Pix is a modular graphics design tool for an Atari 400/800/1200 with one disk drive and a joystick.

While many graphics utilities allow you to draw your own pictures, *Pix* has "libraries" of pictures. Those who are not skilled artistically can create intricate designs by assembling pre-drawn elements into a larger picture. Each module is designed for a slightly different application, and three are included with this issue. The first, called The Landscaper, is an educational aid for children as young as three, who are just learning the initial consonant sounds. The Adventure Screen Maker creates graphics screens, which can be used in arcade games in general, but is best suited for adventures. The third module creates art deco title screens and designs for your software.

All the *Pix* screens use the Atari character graphics modes with modified character fonts. I created the fonts using the DataSoft *Graphic Generator™* utility (with their permission). It is one of the finest graphics utilities available for the Atari 400/800 systems.

Sample Butterfly

After the *Pix* title screen disappears you can choose among several menu options. I have included samples to show what *Pix* can do, so select menu option four: Load a Picture. A program called READER.PIX loads, which reads picture files and displays them on the screen. The first prompt is INPUT PICTURE NAME. The reply is D:BFLY.PIC, which loads a picture I drew with the Landscaper educational module. *Do not include the usual quotes* around any command within *Pix* or the program will crash. Typing only BFLY.PIC also crashes the program, so be careful.

The next prompt, INPUT FONT, requests the font that you wish to use. In this case the proper response is D:EDGAME.FNT. After you have seen this picture get back to the title page by typing RUN "D:TITLE" or by typing RUN "D:READER.PIX". When you have returned to the Reader Program type D:DECO.PIC to the first prompt and D:DECO1.FNT to the second. To see the adventure screen, get into the Reader Program and type D:ADVENT.PIC to the first prompt and D:ADVENT.FNT to the second. All the *Pix* fonts end with the file name extension .FNT and all of the *Pix* sample pictures end with the file name extension .PIC.

You can change the colors you see on your screen. Line 5 of the Reader Program contains the GRAPHICS and SETCOLOR statements. If you don't know how to change the colors in the Atari text modes, this documentation provides a quick review at the end.

Creating PIXures

To make a picture using the *Pix* modules, run the title program and choose Title Page Maker. The program asks you to name the picture you will be working on this session. Let's call the picture SAMPLE. In reponse to the prompt, type D:SAMPLE and press RETURN. *Pix* loads the appropriate font into memory via a machine language subroutine. The screen goes blank for a few seconds and then a border surrounds the screen.

Next a butterfly-shaped cursor appears at the upper left hand corner of the screen. Move it to the right with your joystick (plugged into port 1) and become familiar with its movements on the screen. Each move represents a space where you can place a letter or picture. Move the butterfly toward the middle of the screen and hit the trigger, locking the butterfly into position. Now go to the keyboard and type the letter Q, which appears on the screen in the butterfly's position, leaving the butterfly free to move again. To erase the Q, position the butterfly over it and hit the space bar.

When manipulating the butterfly, be careful when you hit the trigger. Pressing it accidentally locks the butterfly on the screen. If this happens on top of a letter you do not want erased, simply retype the letter and the butterfly will move on. If the butterfly gets stuck on a blank space, just press the space bar to free it up.

Now experiment with the keyboard. All the letters and numbers appear in an art deco style, and I have provided borders and flowers in that style to enhance your title page as well. The documentation contains a description of what is under each key. A copy of this information tacked near your computer helps make things easier for future sessions.

- To start a new screen, type CTRL-C for clear. This erases the present picture from memory.
- To change the background shade, type CTRL-B.
- To change the color of the upper case letters, type CTRL-L.

Note: You must press the joystick trigger, locking the butterfly, before you type any of these commands.

Other color changes result from the various Atari text graphics modes commands. If you hit the trigger and then the Caps Lowr key, the letters print in another color. The trigger, and the Atari logo key in upper case, provide another color. Finally, the trigger and Atari logo key, then the trigger and the lower case key, produce still another color. In all, *Pix* allows you to draw screens in five colors, four for the text and one for the background.

Saving PIXures

Now that you've created the title page for your (soon to be) award-winning program, you'll want to save it. Make sure neither the lower case key nor the Atari logo key is toggled. Now press the trigger and CTRL-K. Your title page turns into normal Atari text and the computer appears to stall for about seven seconds. The disk drive light glows while the program saves your picture under the name you gave it at the start of the session.

The extremely compact disk file for the picture uses only four or five sectors, and the technique is simple: The program turns the screen into one long string and stores it in a text file. When you load the stored picture from disk, the Reader Program converts the string from the file and prints it as a picture on the screen.

DV BONUS

DV BONUS

To see your picture, load the Reader Program and, in response to the prompts, type your picture name (D:SAMPLE in this case) for the first prompt, and then type D:DECO1.FNT to the second prompt.

Because *Pix* does not store any color information, it reverts to the default colors when you reload a picture. Now let's find out how to change them. In the text modes under Atari BASIC, the SETCOLOR statement controls the colors of the characters printed. Color register zero controls the upper case letters, numbers and special characters. To change the color of the upper case letters to orange, the proper command is SETCOLOR 0,15,8. The fifteen refers to the color orange and the eight sets the luminance at a medium shade. (Refer to the appropriate Atari manuals for color combination charts.) The number two color register controls the upper-case inverse letters. The number one color register controls the normal lower case characters, and the number three color register controls the lower case, inverse colors. The number four color register sets the background color register. If you wish to change these values in either the Reader Program or the main program, change line five accordingly, but do not change the GRAPHICS 17 command, only the SETCOLOR commands.

Kid PIX

The Landscaper program is the first of several modules intended to teach young children the beginning consonant sounds. Parents of young children can spend many fun-filled hours teaching them computer skills and word sounds while allowing them to exercise their creativity. The Landscaper program is slightly different from the other two modules because what is on the keyboard and what is printed on the screen do not always correspond. Load the Landscaper program and define your picture name. As you and your child explore the keyboard, emphasize the letter sounds by saying, T is for tree, C is for cloud, and so on. See the reference chart for the full command set. Future *Pix* modules for young children will include animal mix-ups, farm scenes and more environment themes.

PIX Adventurer

One of the fringe benefits of *Pix* is its stinginess with memory. One *Pix* picture takes up a mere 674 bytes in GRAPHICS 1 and a mere 424 bytes in GRAPHICS 2, so a disk can store about 150 PIXures. Suppose each one of those pictures were a different level of a maze or a different scene in an adventure. Using your *Pix* adventure maker, you're not too far away from designing really complex adventure games. Briefly, you can use page-flipping techniques or read in one screen at a time from the disk, or both. In a future article, I will explain in greater detail how to accomplish it. Using the *Pix* adventure scene creator and the predefined characters for transportation, shelter, protection, food and survival, you can lay out an endless variety of fantasy lands and adventures. A complete list of characters is in the Quick Reference section. Note that CTRL-K is the command which stores these pictures.

Future *Pix* modules will include a music processing system which reads music as if it were text, more graphics utilities and fonts, and several educational activities. The latter are my favorite because they let children exercise their creativity without needing sophisticated fine motor skills or artistic ability.

PIX Quick Reference Chart

The Landscaper Module

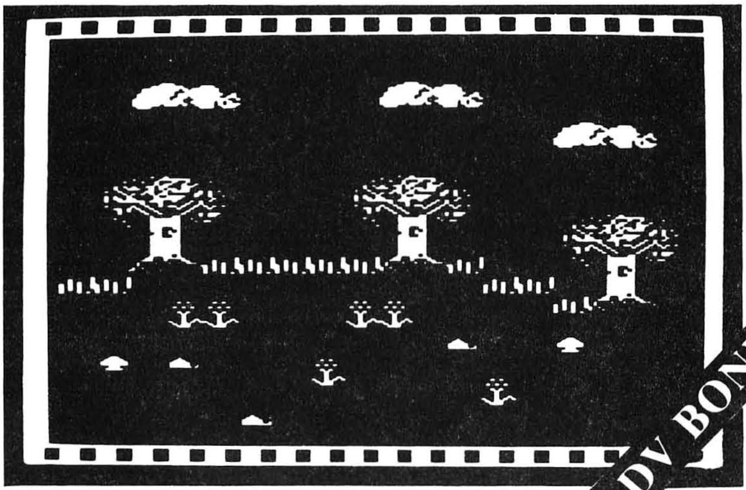
C — Cloud	L — Lighter
D — Darker	M — Mushroom
F — Flower	Q — Quit and start again
G — Grass	S — Snail
K — Keep, or save to disk	T — Tree

Title Page Maker

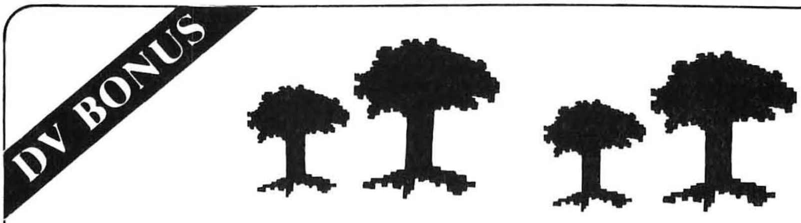
A — Z, 1 — 0 Same as what appears on the key except art deco style font
! — Same character as border
' — Quote marks
# — Slanted lines
$ — Little flower
% — Border, one thick line, over one thin line (top border)
& — Same as % only right border
' — Same as % only left border
● — Same as % only bottom border

All the keyboard symbols, such as parentheses, the hyphen, asterisk and equals symbol translate into their equivalents in the new font. However, the following generate special graphics characters:

, — Upper left side of small flowery frame
[— Lower-left side of small flowery frame
. — Upper-right side of small flowery frame
] — Lower-right side of small flowery frame
/ ? : ; — Four parts of a philodendron vine
CTRL-B — Change background shade
CTRL-L — Change upper case letter color
CTRL K — Save to disk



PIX



Adventure Screen Generator

A	Tree	O	Jet
B	Grass	P	Dollar sign (money)
C	Wave/Water	Q, R	Shelter (space)
D, E	Mountain	S	Tie fighter
F, G	Bridge	T	Musical note
H	Snow/Dirt/Magic Powder	U	Race car left
I	Boulder	V	Race car right
J	Horizontal road	W	Moon rover
K	F sign (food/fuel)	X	Regular car left
L	Old boat	Y	Regular car right
M	Skull and crossbones (danger)	Z	Rocket/missile
N	Question mark (take a chance)	,	Flower

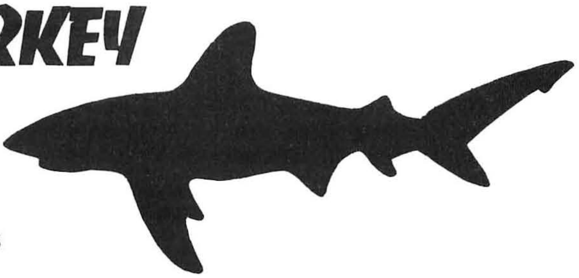
.	Mushroom
[	Lance/Sword
]	Shield
/	Laser/Gun
?	Magic wand
:	Key
;	Potion
½	Building
*	Space hut
±	Barrier/brick wall
-	Mean creature
=	Friendly computer
+	Castle
—	Diamond
!	Symbol for man
#	Woman
$	Arrow pointing up
%	Down pointing arrow
&	Right arrow
'	Left arrow
●	Sun shining
(	Crater
)	Vertical road
<	Race car up

CTRL-B	change background shade
CTRL-L	change upper-case color
CTRL-C	clear and start again
CTRL-K	keep and save to disk

Notes:

After saving to disk, press Reset to clear Player/Missile garbage. Toggling the lower case or Atari logo key will change character color on the Title Page Maker and Adventure Screen Maker.

SHARKEY



by Dale Thoms

Sharkey is an arcade-type game for an Atari® with 24K RAM (32K with disk), and a joystick.

Looking at the world from the shark's point of view, you move through a coral reef, eating rare tropical fish as you go. The white fish are worth twenty points each, but avoid the dark green fish which inhabit the center of the screen — they are poisonous. In the same area, you encounter spears, shot at random intervals. At the bottom of the screen, in the seaweed, you see eels, which can also spell your untimely demise.

While watching out for all of those dangers, be careful not to bump into any coral. If you do, you will be sent back to the box in the center of the screen. Keep an eye on the clock. If the time expires, so do you. The number at the right of the score display is the number of sharks you have left in addition to the one in play.

Program Operation

PMBLANK is the machine language Vertical Blank Interrupt (VBI) routine which handles the movement of the shark, the spears and the eels. VBIs occur every sixtieth of a second, and instruct the 6502 microprocessor to stop the program it is executing and jump to the VBI processing routine located in the Operating System (OS). This routine increments the POKEY timers, interprets controller input, and copies data from shadow registers in memory to the hardware registers. If the OS routine has not been instructed to jump to a user supplied routine, the 6502 picks up where it left off in the main program, oblivious to the fact that a VBI occurred. If a user routine such as PMBLANK is used, the 6502 returns to the main program after the routine has finished executing. Putting the Player graphics movement routines into the VBI sequence allows them to run simultaneously with the BASIC program.

The first part of PMBLANK operates as follows: The BASIC program reads the joystick and POKEs the value into memory location 1789 ($06FD Hex). PMBLANK looks at this location and moves the shark in the indicated direction. Six of the eight possible joystick positions cause the shark image to change. Player 0 is modified with the appropriate bytes contained in the Player image area which is located at page PM + 5. The other two directions, up and down, use the existing shark image. PMBLANK checks location 1787 ($06FB) to see if the joystick button is pressed. If it is, then PMBLANK will move the shark once every VBI; otherwise the shark moves once every two VBIs.

The second part of PMBLANK moves the eels and spears. The BASIC program POKEs the chosen number of the movements into location 1785 ($06F9) for spears or location 1781 ($06F5) for eels. When PMBLANK sees that a number has been POKEd into the eels' location, it positions Player 1 at the left edge of the screen and moves it two positions to the right every VBI, and decrements location 1781. When this value reaches 0, Player 1 is removed from the screen.

The routine for spear movement is similar with a few exceptions: Players 2 and 3 are positioned inside the central box on the left and right edges. Spears do not traverse the entire screen; they stop at the opposite end of the central box. Spears are moved three positions every VBI.

Variables

A,B: Used for loops, etc.
C: Dummy variable used for calling machine language routines.
CMX: Maximum value of COUNT.
COL1: Used for collision detection.
COUNT: Number of times the main loop has been run.
DLIST: Address of the beginning of the Display List.
E$: Contains a machine language routine to zero out PMG RAM.
EMX: Number of times PMBLANK should move eels.
FISH: Number of white fish the shark has eaten.
H0: Horizonal position of the shark computed to a character position.
MAX: Number of white fish at the start of each screen.
PM: Starting address of PMG RAM (4.25K or 17 pages below RAMTOP).
SCORE: Player's total score.
SCREEN: Number of current screen (used as a difficulty level).
SHARK: Number of remaining sharks excluding the one in play.
SMX: Number of times PMBLANK should move the spears.
START: Starting address of redefined character set in RAM.
T0,T1: Used to keep track of the game timer.
TIME: Amount of time at the start of each screen.
V0: Vertical position of the shark computed to a character position.
X,Y,Z: Used for reading in the character set data.
XFR$: Contains a machine language routine to download the ROM character set.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS         Atari BASIC         SS
SS          'Sharkey'          SS
SS      Author: Dale Thoms     SS
SS      Copyright © 1983       SS
SS SoftSide Publications, Inc  SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

If you don't wish to type this program, it is available on Issue #45 SoftSide DV and CV.

Initialization.

```
10 GOSUB 9000:GOSUB 10000:GOSUB 11000
20 GOSUB 14000:GOSUB 20000
30 GOSUB 21000:GOSUB 24000:GOSUB 12000
40 GOSUB 25000:GOSUB 23000
```

Main game routine.

Clear collision register and POKE the joystick position into a page six memory location so that PMBLANK can move the shark.

```
100 POKE 53278,1:POKE 1789,STICK(0)
```

Start movement of spears and eels by POKEing a value into page six for PMBLANK. POKE 1781,EMX tells PMBLANK to move the eels (Player 1) 83 times (all the way across the screen). POKE 1785,SMX moves the spears (Players 2 and 3) 31 steps across the central playfield.

```
110 COUNT=COUNT+1
120 IF COUNT=15 THEN POKE 1781,EMX
130 IF COUNT=60 THEN POKE 1785,SMX
140 IF COUNT>CMX THEN COUNT=0
```

Tell PMBLANK to double the shark speed if the trigger is pressed.

```
150 POKE 1787,STRIG(0)
```

Shark wrap around routine.

```
160 IF PEEK(1790)>204 THEN POKE 1790,4
5
170 IF PEEK(1790)<44 THEN POKE 1790,20
3
```

Check for collision.

```
180 IF PEEK(53252)>0 THEN GOSUB 5000
190 IF PEEK(53260)>0 THEN GOSUB 6000
```

Keep track of game time.

```
200 T0=T0+1
210 IF T0=15 THEN T0=0:T1=T1-1:POKE 65
6,0:POKE 657,32:? "  ":POKE 656,0:POKE
 657,32:PRINT T1;
220 IF T1=0 THEN GOSUB 6000
```

Clear the "Attract Mode" counter and return to beginning.

```
230 POKE 77,0:GOTO 100
```

Collision Routines.

If shark has collided with the playfield graphics, GOSUB the appropriate routine.

```
5000 COL1=PEEK(53252):IF COL1>=8 THEN
COL1=COL1-8
5010 IF COL1>=4 THEN COL1=COL1-4:GOSUB
 5100
5020 IF COL1>=2 THEN COL1=COL1-2:GOSUB
 5300
5030 IF COL1=1 THEN GOSUB 5500
5040 POKE 53278,1:RETURN
```

Collision with white fish.

Convert the shark's position into a character position. Check that position for a white fish. If it's there, erase it, and start the beep sound.

```
5100 H0=(PEEK(1790)-48)/8:V0=(PEEK(179
1)-32)/8
5110 LOCATE H0,V0,A:IF A<>202 THEN RET
URN
5120 SOUND 0,25,10,10:POSITION H0,V0:?
 #6;" "
```

Add 20 points to SCORE, check for end of play on current screen, and return if the screen is not clear.

```
5130 SCORE=SCORE+20:POKE 656,0:POKE 65
7,6:? SCORE
5140 FISH=FISH+1:IF FISH<MAX THEN SOUN
D 0,0,0,0:RETURN
```

Erase shark, play end of screen music, add bonus score for time remaining on the clock, and GOSUB 13000 to set up the next screen.

```
5150 POKE 1789,15:POKE 1788,64:SOUND 0
,0,0,0
5160 POKE 53278,1:SOUND 0,0,0,0
5170 GOSUB 5180:GOSUB 13000:RETURN
5180 SOUND 1,121,10,12:GOSUB 5210:SOUN
D 1,96,10,12:GOSUB 5210:SOUND 1,108,10
,12:GOSUB 5210
5190 SOUND 1,91,10,12:GOSUB 5210:SOUND
 1,96,10,12:GOSUB 5210:SOUND 1,81,10,1
2:GOSUB 5210
5200 SOUND 1,60,10,12:FOR A=0 TO 80:NE
XT A:SOUND 1,0,0,0:GOSUB 5210:GOTO 522
0
5210 FOR A=0 TO 60:NEXT A:SOUND 0,0,0,
0:FOR A=0 TO 10:NEXT A:RETURN
5220 SCORE=SCORE+10*T1:POKE 656,0:POKE
 657,6:? SCORE:POKE 656,0:POKE 657,32:
? "  ":RETURN
```

Collision with walls.

Stop shark's movement.

```
5300 POKE 1789,15
```

Play coral collision music.

```
5310 POKE 709,80:FOR A=25 TO 60:SOUND
0,A,10,8:NEXT A:POKE 709,94:FOR A=42 T
O 77:SOUND 0,A,10,8:NEXT A
5320 POKE 709,80:FOR A=59 TO 92:SOUND
0,A,10,8:NEXT A:POKE 709,84:FOR A=0 TO
 15:NEXT A:SOUND 0,0,0,0
```

Erase shark and reset its position to the central box.

```
5330 POKE 1788,64:FOR A=0 TO 100:NEXT
A:POKE 1790,124:POKE 1791,105:POKE 178
8,8
```

Play more shark music.

```
5340 SOUND 0,50,10,8:FOR A=0 TO 30:NEX
T A:SOUND 0,37,10,8:FOR A=0 TO 30:NEXT
 A:SOUND 0,24,10,8:FOR A=0 TO 30
5345 NEXT A:SOUND 0,0,0,0
5350 FOR A=0 TO 100:NEXT A
5360 FOR A=0 TO 2:SOUND 0,15,10,15:FOR
 B=0 TO 35:NEXT B:SOUND 0,0,0,0:FOR B=
0 TO 85:NEXT B:NEXT A
5370 RETURN
```

Collisions with dark green fish and seaweed.

If the shark's vertical position is less than 160 (above the seaweed), it collides with a poisonous fish.

```
5500 IF PEEK(1791)<160 THEN GOSUB 6000
:RETURN
5510 RETURN
```

Shark death routine.

Stop the shark's motion and turn it upside down.

```
6000 POKE 1789,15:POKE 1788,0
```

Play the dead shark music.

```
6010 SOUND 0,45,10,8:FOR A=0 TO 100:NE
XT A:SOUND 0,0,0,0:SOUND 0,47,10,8:FOR
 A=0 TO 100:NEXT A:SOUND 0,0,0,0
6020 SOUND 0,50,10,8:FOR A=0 TO 150:NE
XT A:SOUND 0,0,0,0:SOUND 0,53,10,8:FOR
 A=0 TO 50:NEXT A:SOUND 0,0,0,0
6030 SOUND 0,50,10,8:FOR A=0 TO 100:NE
XT A:SOUND 0,0,0,0
6040 FOR A=0 TO 200:NEXT A
```

Move the shark to the top of the screen.

```
6050 B=PEEK(1791):FOR A=B TO 32 STEP -
1:SOUND 0,188-A,10,2:POKE 1791,A:NEXT
A
```

If any sharks remain, return to main routine.

```
6060 SOUND 0,0,0,0:SHARK=SHARK-1
6070 IF SHARK>-1 THEN T1=TIME:POKE 656
,0:POKE 657,18:? SHARK:RETURN
```

Set up a new game.

```
6080 POKE 1788,64:POP :FOR A=0 TO 250:
NEXT A
6090 POKE 656,0:POKE 657,23
6100 ? "PUSH fire TO"
6110 POKE 656,1:POKE 657,4:? "START OV
ER"
6120 IF STRIG(0)<>0 THEN 6120
6130 SCREEN=1:SHARK=2:SCORE=0:? CHR$(1
25):POKE 656,1:POKE 657,0:? " one mome
nt please"
6145 GOSUB 7000
6150 GOTO 20
```

Preliminary game initialization.

Call a machine language routine to zero out the Player RAM area.

```
7000 FOR A=0 TO 3:C=USR(ADR(E$),(PM+13
+A)*256):NEXT A:RETURN
```

Reserve 4.25K of RAM and set up the title display.

```
9000 RESTORE :DIM E$(19):FOR X=1 TO 19
:READ D:E$(X)=CHR$(D):NEXT X:PM=PEEK(1
06)-17:POKE 106,PM:GOSUB 7000
9005 DATA 104,104,133,213,104,133,212,
169,0,160,0,145,212,200,192,255,208,24
9,96
9010 GRAPHICS 1:DLIST=PEEK(560)+PEEK(5
61)*256
9020 POKE DLIST+10,7:POKE DLIST+17,2:P
OKE 710,1:POKE 752,1
9030 POSITION 7,5:? #6;"sharkey":POSIT
ION 0,12:? #6;"Written and produced by
-":POSITION 6,15:? #6;"DALE THOMS"
9040 ? CHR$(125);CHR$(127);"Please wai
t 17 seconds for":? CHR$(127);"initial
ization."
9050 RETURN
```

POKE in the redefined character set using Alan J. Zett's machine language routines.

```
10000 START=(PEEK(106)+1)*256
10010 DIM XFR$(37):FOR X=1 TO 37:READ
D:XFR$(X)=CHR$(D):NEXT X
10020 Z=USR(ADR(XFR$))
10030 READ X:IF X=-1 THEN RETURN
10040 FOR Y=0 TO 7:READ Z:POKE X+Y+STA
RT,Z:NEXT Y:GOTO 10030
10050 DATA 104,169,0,133,203,133,205,1
69,224,133,206,165,106,24,105,1,133,20
4,0,177,205,145,203,200,208,249,230
10060 DATA 204,230,206,165,206,201,228
,208,237,96
10100 DATA 512,0,0,0,0,0,0,0,0
10101 DATA 576,255,255,7,14,28,56,112,
224
10102 DATA 592,255,255,224,112,56,28,1
4,7
10103 DATA 776,0,64,192,193,199,108,60
,48
10104 DATA 784,99,51,51,31,28,56,48,48
10105 DATA 792,7,206,204,110,198,198,1
03,99
10106 DATA 800,27,155,179,182,182,230,
119,119
10107 DATA 808,24,50,179,227,97,97,59,
27
```

```
10108 DATA 816,110,108,204,198,204,92,
124,192
10109 DATA 824,195,246,54,99,195,199,1
10,110
10110 DATA 832,153,141,205,206,198,102
,102,60
10111 DATA 840,145,219,206,108,204,201
,153,153
10112 DATA 848,0,0,0,88,60,88,0,0
10113 DATA 528,7,7,3,7,7,3,7,7
10114 DATA 536,7,7,3,7,7,223,255,255
10115 DATA 552,255,255,223,7,7,3,7,7
10116 DATA 584,0,0,0,0,7,15,15,15
10117 DATA 600,15,15,15,7,0,0,0,0
10118 DATA 608,240,240,240,224,0,0,0,0
10119 DATA 616,255,255,219,0,0,0,0,0
10120 DATA 624,0,0,0,0,0,219,255,255
10121 DATA 632,0,0,0,0,224,240,240,240
10122 DATA 648,255,255,251,224,224,192
,224,224
10123 DATA 688,224,224,192,224,224,192
,224,224
10124 DATA 720,224,224,192,224,224,251
,255,255,-1
```

POKE in the machine language PMBLANK routine.

```
11000 POKE 559,62:POKE 53277,3:POKE 54
279,PM+9
11010 POKE 704,72:POKE 53248,100
11020 FOR A=0 TO 218:READ B:POKE 1536+
A,B:NEXT A
11100 DATA 173,251,6,201,0,240,20,173,
250,6,201,0,208,8,169,1
11110 DATA 141,250,6,24,144,87,169,0,1
41,250,6,173,253,6,201,14
11120 DATA 240,84,201,7,240,86,201,13,
240,93,201,11,240,95,201,6
11130 DATA 240,102,201,5,240,112,201,9
,240,122,201,10,240,53,160,8
11140 DATA 173,252,6,133,203,173,255,6
,201,188,208,6,206,255,6,173
11150 DATA 255,6,133,205,169,0,145,205
,136,177,203,145,205,192,0,208
11160 DATA 247,198,205,169,0,145,205,1
73,254,6,141,0,208,76,98,228
11170 DATA 24,144,203,24,144,76,206,25
5,6,24,144,194,238,254,6,169
11180 DATA 8,141,252,6,24,144,183,238,
255,6,24,144,177,206,254,6
11190 DATA 169,24,141,252,6,24,144,166
,206,255,6,238,254,6,169,32
11200 DATA 141,252,6,24,144,152,238,25
5,6,238,254,6,169,40,141,252
11210 DATA 6,24,144,138,238,255,6,206,
254,6,169,48,141,252,6,24
11220 DATA 144,174,206,255,6,206,254,6
,169,56,141,252,6,24,144,160
11230 DATA 162,6,160,0,169,7,32,92,228
,104,96
11240 SCREEN=1:SHARK=2:SCORE=0:RETURN
```

Screen set-up routine.

Print the game board and initialize some variables to the first level of difficulty values.

```
12000 POSITION 0,0
12005 REM IN LINES 12010-12180, LOWER
        CASE REPRESENTS CTRL CODES.
12010 ? #6;"qmmmmmmmmmmmmmmmmmme";
12020 ? #6;"vJ J J  J  J  J J Jb";
12030 ? #6;"v io io inno io io b";
12040 ? #6;"vJklJklJkeqlJklJklJb";
12050 ? #6;"v    J  JbvJ  J    b";
12060 ? #6;"v innno iczo innno b";
12070 ? #6;"v kmmmlJkmmlJkmmml b";
12080 ? #6;"v  J J  J  J  J J  b";
12090 ? #6;"v  innnnnnnnnnno  b";
12100 ? #6;"vJ bqmmme  qmmmev Jb";
12110 ? #6;"v Jbv  J       JbvJ b";
12120 ? #6;"v Jbv J   J  J bvJ b";
12130 ? #6;"vJ bvJ  J       bv Jb";
12140 ? #6;"v  bv       J  Jbv  b";
12150 ? #6;"vJ bvJ J J J   bv Jb";
12160 ? #6;"vJJbv  J       J bvJJb";
12170 ? #6;"v  bv innnnnno bv  b";
12180 ? #6;"v Jkl kmmmmmml klJ b";
12185 REM LINES 12190-12200 DO NOT
        CONTAIN CTRL CODES. TYPE
        LOWER CASE AS NORMAL.
12190 ? #6;" CJe gJIeGg CeJgJEJi";
12200 ? #6;"aBAdJfAHdFfJBdafaDAh";
12210 FISH=0:MAX=59:TIME=100:CMX=150:S
MX=31:EMX=83
```

Increase the difficulty level based on the value of SCREEN.

```
12220 IF SCREEN<2 THEN 12990
12230 POSITION 10,12:? #6;"J":POSITION
 13,12:? #6;"J":TIME=90:CMX=120
```

SHARKEY

```
12240 POSITION 0,5:? #6;CHR$(26);CHR$(
14);CHR$(14);:POSITION 17,5:? #6;CHR$(
14);CHR$(14);CHR$(3);
12245 POSITION 0,6:? #6;CHR$(17);CHR$(
13);CHR$(13);:POSITION 17,6:? #6;CHR$(
13);CHR$(13);CHR$(5);
12250 IF SCREEN<3 THEN 12990
12260 A=(PM+15)*256+154:POKE A,4:POKE
A+1,194:POKE A+2,127:POKE A+3,194:POKE
 A+4,4:TIME=80
12270 POSITION 9,0:? #6;CHR$(5);CHR$(1
7);:POSITION 9,1:? #6;CHR$(2);CHR$(22)
;:POSITION 9,2:? #6;CHR$(3);CHR$(26);
12280 IF SCREEN<4 THEN 12990
12290 TIME=70:CMX=100:A=(PM+14)*256+18
6:POKE A,96:POKE A+1,147:POKE A+2,12
12300 POSITION 0,10:? #6;CHR$(26);CHR$
(15):POSITION 18,10:? #6;CHR$(9);CHR$(
3)
12302 POSITION 0,11:? #6;CHR$(17);CHR$
(12):POSITION 18,11:? #6;CHR$(11);CHR$
(5)
12305 POSITION 2,13:? #6;CHR$(9);CHR$(
3):POSITION 16,13:? #6;CHR$(26);CHR$(1
5)
12306 POSITION 2,14:? #6;CHR$(11);CHR$
(5):POSITION 16,14:? #6;CHR$(17);CHR$(
12)
12307 POSITION 0,16:? #6;CHR$(26);CHR$
(15):POSITION 18,16:? #6;CHR$(9);CHR$(
3)
12308 POSITION 0,17:? #6;CHR$(17);CHR$
(12):POSITION 18,17:? #6;CHR$(11);CHR$
(5)
12310 TIME=60:CMX=70:POSITION 11,10:?
#6;"J":POSITION 7,12:? #6;"J"
12320 IF SCREEN<6 THEN 12990
12330 TIME=50:CMX=65:POKE (PM+9)*256+3
1,232:POKE (PM+9)*256+45,202:POKE (PM+
9)*256+78,232:SMX=23:EMX=55
```

Give the player three warning beeps before returning control of the shark.

```
12990 FOR A=0 TO 2:SOUND 0,15,10,15:FO
R B=0 TO 15:NEXT B:SOUND 0,0,0,0:FOR B
=0 TO 25:NEXT B:NEXT A:T1=TIME:RETURN
```

Reset the shark's position to the central box, increment the SCREEN variable, reset some variables, and call the screen set-up routine.

```
13000 FOR A=0 TO 200:NEXT A:POKE 1790,
124:POKE 1791,105:POKE 1788,8
13020 SCREEN=SCREEN+1:FISH=0:T0=0:GOSU
B 12000
13030 COL1=0:RETURN
```

POKE spear and eel shapes into PMG RAM.

```
14000 RESTORE 14030
14010 FOR A=0 TO 4:READ B:POKE (PM+15)
*256+120+A,B:NEXT A
14020 FOR A=0 TO 4:READ B:POKE (PM+16)
*256+138+A,B:NEXT A
14030 DATA 4,194,127,194,4,32,67,254,6
7,32
14040 A=(PM+14)*256+180:POKE A,96:POKE
 A+1,147:POKE A+2,12
```

Initialize the page six locations used by PMBLANK.

```
14050 RETURN
20000 POKE 1790,124:POKE 1791,105
20010 POKE 1789,0:POKE 1788,8
20020 POKE 1787,1:POKE 1786,0
20030 POKE 206,PM+13:POKE 205,0:POKE 2
04,PM+5:POKE 203,0
20040 A=0
20050 FOR A=0 TO 63:READ B:POKE (PM+5)
*256+A,B:NEXT A
20060 RETURN
20070 DATA 121,254,121,24,12,0,0,0
20080 DATA 48,24,158,127,158,0,0,0
20090 DATA 0,0,0,0,0,0,0,0
20100 DATA 12,24,121,254,121,0,0,0
20110 DATA 48,24,94,127,158,0,0,0
20120 DATA 48,24,158,127,94,0,0,0
20130 DATA 12,24,121,254,122,0,0,0
20140 DATA 12,24,122,254,121,0,0,0
```

Set up the colors for spears and eels, and POKE in the PMBLANK VBI routine.

```
21000 POKE 705,156:POKE 706,56:POKE 70
7,56
21010 POKE 1646,0:POKE 1647,PM+9
21020 FOR A=0 TO 90:READ B:POKE (PM+9)
*256+A,B:NEXT A
21030 RETURN
21040 DATA 173,249,6,201,0,208,19,141,
2,208,141,3,208,169,77,141
```

```
21050 DATA 247,6,169,171,141,248,6,24,
144,29,174,247,6,232,232,234
21060 DATA 232,142,247,6,142,2,208,174
,248,6,202,202,202,234,142,248
21070 DATA 6,142,3,208,206,249,6,173,2
45,6,201,0,208,11,141,1
21080 DATA 208,169,42,141,246,6,24,144
,15,174,246,6,232,232,234,142
21090 DATA 246,6,142,1,208,206,245,6,7
6,98,228
```

Set up the score board area.

Change the text window to GRAPHICS 1 mode and create a Display List Interrupt to display uppercase text instead of graphics symbols.

```
23000 DLIST=PEEK(560)+256*PEEK(561)
23010 POKE DLIST+24,134:POKE DLIST+25,
70
23020 POKE DLIST+28,6:POKE DLIST+29,6:
POKE DLIST+30,6
23030 FOR A=0 TO 17:READ B:POKE (PM+10
)*256+A,B:NEXT A
23040 DATA 72,138,72,174,244,2,202,202
,141,10,212,142,9,212,104,170,104,64
23050 POKE 512,0:POKE 513,PM+10
23060 POKE 54286,192
```

Print headings in the text box.

```
23070 ? CHR$(125)
23080 POKE 656,0:POKE 657,0:? "score":
POKE 656,0:POKE 657,27:? "time":POKE 6
56,0:POKE 657,18:? SHARK
23090 RETURN
```

Initialize the rest of the colors, and switch the character set pointer from the old ROM set to the redefined RAM set.

```
24000 POKE 708,182:POKE 709,84:POKE 71
0,110:POKE 711,214:POKE 712,160:POKE 6
23,8
24010 POKE DLIST+10,6:POKE DLIST+17,6:
? #6;CHR$(125):POKE 756,START/256+2
24020 RETURN
```

Initialize the PMBLANK VBI routine.

```
25000 C=USR(1744):RETURN
```

STOMP TABLE

For **ATARI® SHARKEY**

(Modified Parameters:
Line count = 5, byte count = 300)

LINES	STOMP CODE	LENGTH
10 - 100	CI	145
110 - 150	NA	119
160 - 200	CX	140
210 - 5010	PA	207
5020 - 5110	DR	166
5120 - 5160	EU	203
5170 - 5210	PJ	320
5220 - 5330	SK	343
5340 - 5370	TT	228
5500 - 6020	MQ	265
6030 - 6070	JB	228
6080 - 6120	HS	142
6130 - 9000	JO	254
9005 - 9040	CH	359
9050 - 10030	IT	141
10040 - 10101	YD	275
10102 - 10106	UP	198
10107 - 10111	EU	210
10112 - 10116	YK	158
10117 - 10121	KX	171
10122 - 11010	LZ	206
11020 - 11130	GQ	287
11140 - 11180	VL	320
11190 - 11230	NR	295
11240 - 12030	XI	148
12040 - 12080	KY	160
12090 - 12130	JE	160
12140 - 12180	IR	160
12190 - 12230	UR	201
12240 - 12270	GX	398
12280 - 12305	DH	314
12306 - 12320	NL	307
12330 - 13030	CC	317
14000 - 14040	QK	211
14050 - 20030	ZC	144
20040 - 20080	ET	129
20090 - 20130	MG	150
20140 - 21030	VZ	156
21040 - 21080	ZB	316
21090 - 23030	HN	211
23040 - 23080	RR	223
23090 - 25000	NB	186

SoftSide®

ADVENTURE SERIES

**Issue 45 Adventure:
Jack The Ripper II**

Did the evil Ripper ever really die? You are an inspector at Scotland yard. Can you stop the Ripper before he strikes again?

SoftSide Adventure Series SoftSide CV SoftSide DV

What would you say to a program that asks, "What do you want to do?" Well, you might say, "GET APPLE" or "KILL SPIDER", because that's how the *SoftSide Adventure Series* works.

Each issue, the latest Adventure takes you to another world of fantasy, puzzles and thrills. Your first task — survival — can be daunting until you figure out the *right* way to do it. Each adventure sets a fundamental goal before you. You might have to rescue a princess, or depose a wicked ruler. Along the way, dozens of subsidiary tasks beset you. You'll have to be ingenious and persevering, and your rewards will be great.

To "win" a fantasy/adventure game, you must solve the author's devious puzzles, and overcome the obstacles that confront you — whether they be dragons or desperadoes. Death, should it come, is transitory — just re-run the program to live again!

Express your wishes with one- or two-word commands, like "LOOK", "NE" (for northeast), or "GET FROG". Use "I" to get an inventory of your possessions. The introduction to each Adventure explains this more fully.

Winning an Adventure is hard enough without the additional requirement that you do it all at once. In recognition of this, the Disk Version of the *SoftSide Adventure Series* now features two new commands: "SAVE GAME" and "LOAD GAME".

To start up the Adventure, just run the program called "INTRO", "INTRO/BAS", or "INTRO.BAS" on your disk, or select the Adventure from the DV menu. On cassette, the INTRO program is the one just before the Adventure.

The Adventure runs in any Atari® with at least 32K RAM (40K disk).

Here are the encrypted hints for *Mad House,* the Adventure in Issue 44.

Silver dollar: WL MLG KFG RM EVMWRMT NZXSRMV FMGRO TREVM GL TVLITV DZHSRMTGLM LI UORKKVW YB OFXPB XSFXPB.
Top hat: WL MLG TREV GL NZTRXRZM FMGRO GIRXPB WRXPB SZH YVVM WVZOG DRGS.
Tricky Dicky: WIVHH FK ORPV ZYV ORMXLOM.
Shakespeare: TREV SRN Z ULFI-OVZU XOLEVI.
Bunny: ORPVH XOLEVI.
To get into the garden: HLFMW XZM YIVZP TOZHH.
To pass a day: HOVVK LM BLFI YFMP.

To escape:
A. VMGVI OZFMWIB ILLN LM KRXP FK WZB.
B. SZEV Z ORG UOZHSORTSG.
C. SRWV RM OZFMWIB YZT.
D. OLLP OZFMWIB; GSVM TVG ZMW DVZI GSV FMRULIN.
E. OVZEV YZT ZUGVI GIFXP HGLKH.
F. VCRG WLLI.

Notes

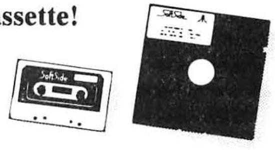
Tired of Typing?

- Use And Enjoy These Programs Right Away!
- Avoid Worry About Typos!
- Get Additional Programs to Enjoy Without Typing.

Order this Issue's Programs on Disk or Cassette!

Send the coupon below to:

SoftSide Publications, Inc.
10 Northern Blvd.
Northwood Executive Park
Amherst, NH 03031



SoftSide CV SoftSide DV

SoftSide® Selections

ORDER THIS ISSUE'S MEDIA

Having difficulty finding the time to input the programs from this issue of *SoftSide* **Selections**? Don't despair — the cassette and disk versions are still available! **And**, each DV (disk version) contains an additional program.

☐ **#45 DV $16.00** ☐ **#45 CV $8.00**

Disk Only: ☐ Apple® ☐ IBM® PC
Disk or Cassette: ☐ Atari® ☐ TRS-80® Mod. I ☐ TRS-80® Mod. III

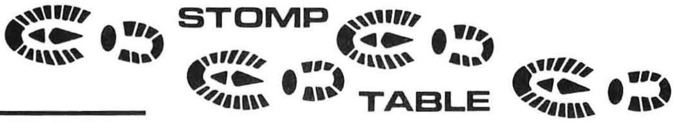
Name ______________________
Address ______________________
City/State ______________ Zip ________
☐ **Check or Money Order**
☐ MasterCard ☐ VISA
Name of Cardholder ______________________
MC #/VISA # ______________________
Exp. Date ___/___/___ Signature ______________

SoftSide Selections
Draw 7 Plus
45
PIX
SHARKEY
FEATURING STOMP
JACK The RIPPER II
A product of SoftSide Publications, Inc. 10 Northern Blvd., Amherst, NH 03031

Prices subject to change without notice. Apple®, IBM®, Atari® and TRS-80® are registered trademarks of Apple Computer, Inc., International Business Machines, Inc., Warner Communications, and Tandy Corporation respectively.

General Information

These are the standard procedures for the programs published by **SoftSide** Publications, Inc. Sometimes, a particular program does not lend itself to these procedures. Always read the specific instructions accompanying a program. They will instruct you if there are any variances from the following procedures. Also, back issues of **SoftSide** Magazine may differ in some details.



STOMP Tables

At the conclusion of each **SoftSide** listing, we include a **STOMP** Table. **STOMP** for the Apple, Atari, IBM PC and TRS-80 appeared in Issue #45. **STOMP** supersedes **SWAT** as **SoftSide's** standard debugging tool to help those who type BASIC programs from the pages of **SoftSide Selections.** If you don't have **STOMP,** we'll send you a free reprint. Send a business-sized, self-addressed, stamped envelope to:

SoftSide Publications, Inc.
Department **STOMP**
10 Northern Blvd.
Northwood Executive Park
Amherst, NH 03031

Be sure to tell us that you have an Atari computer.

Magnetic Media

Disks do not carry the DOS.SYS and DUP.SYS files, and are not "bootable." In order to use your **SoftSide** DV, put a disk with DOS on it, such as a copy of your DOS 2 master disk, into your disk drive, and start up your Atari. The BASIC cartridge must be inserted before you do this. When you see the word READY, insert the **SoftSide Selections** disk, and type this command:

```
RUN "D:COVER"
```

A menu program will then run.

Our disks are in DOS 2.0S format. They should not be used with DOS 1.

Tapes CLOAD in the normal manner. If you encounter difficulty, try this procedure:

- 1. POKE 54018,54
- 2. Turn up the volume on your TV.
- 3. Type CLOAD, and press RETURN once.
- 4. Press the play button, and listen.
- 5. When you hear the steady leader tone, press RETURN

Side two of the tape is a duplicate of side one.

SoftSide Selections disks and tapes are duplicated on reliable, professional equipment. Bad copies are exceedingly rare. Nevertheless, the trip through the mail occasionally results in damage to the sensitive magnetic media. If, after a reasonable number of attempts on well-adjusted, clean equipment, you are unable to load a program, return it to us along with a precise explanation of your problem. We shall send you a replacement.

SoftSide Selections media are not copy-protected. We urge you to make an archival backup of your disk or tape as soon as you receive it, as our replacement policy is valid for only 30 days. Please resist the urge to give away copies of copyrighted material.

Line Listings

The line listings in this booklet are in standard 38-column format, with special conventions for representing unprintable characters:

● You must type underlined characters, including blank spaces, in inverse video.

● When graphics or control characters are included in a string (between quotation marks), a nearby REM statement will make note of it; in such cases, graphics characters appear as the corresponding lower-case letters, and control characters appear as the corresponding unshifted key symbols. For example: The lower-case letter **s** represents a graphic cross, which you type by pressing the S key while holding down the CTRL key; the = sign represents CTRL-down-arrow, which you type by pressing and releasing the ESC key, then pressing the = key while holding down CTRL. For more information about entering control characters, refer to Appendix F and the back cover of your **Atari Reference Manual.**

There are two exceptions to our above convention: A clear-screen character (ESC SHIFT-CLEAR) appears in our listings as a right-hand brace, which looks like this: } . The other exception is that a shifted = sign appears as a broken, vertical line: ¦.

Occasionally, a program will demand that we vary from these conventions. In such a case, a nearby REM statement or the program's introductory article will clearly note the special instructions.

System Requirements

The necessary memory and other equipment you need to run a program are listed in the introductory paragraph of the article for each program. Also see the **SoftSide Adventure Series** elsewhere in this booklet.

The Best of SoftSide

NOW AVAILABLE ON DISK

☐ **YES**...I want to enjoy the best programs — Games, Utilities, Adventures — from SoftSide Magazine's past. Please rush my copies of The Best of SoftSide to:

Name ____________________

Address ____________________

City ____________ State ______ Zip ______

Here's my order For The Best of SoftSide

Computer	Quantity Book Only $14.95 ea.	Quantity Book plus Disk $49.95 ea.
Apple®		
Atari®		
TRS-80®		
TOTAL COPIES	@$14.95	@$49.95

☐ I've already ordered The Best of SoftSide book. I'm enclosing $35 for the disks only.

I'm paying by:

☐ Check # ____________

☐ Money Order # ____________

☐ VISA

☐ MasterCard

Total Enclosed $____________

(Foreign order please include $5 postage PAY IN USA FUNDS ONLY.)

VISA/MasterCard # ____________ Exp. Date ______

Name of Cardholder ____________

Signature ____________

Like **SoftSide**? Wait 'til you've seen our **BEST!**

The Best of SoftSide!

For the past four years, **SoftSide** Magazine has been bringing Apple®, Atari®, and TRS-80® owners the best in BASIC software. But now you can do even better...**The Best of SoftSide.** From all our back issues, we've selected the most useful...the most entertaining...the most fun programs **SoftSide** has ever published.

The Best of SoftSide is available in three versions...one for Apple, Atari, and TRS-80. Each contains page after page of BASIC code for adventures, simulations, practical applications, and much more.

To order your copy of **The Best of SoftSide,** fill out the coupon, and mail it along with $14.95 to **SoftSide,** 10 Northern Blvd., Northwood Executive Park, Amherst, New Hampshire 03031.

SoftSide® Selections

Here's **SoftSide Selections,** the handy, pull-out booklet with program listings for your Atari 400/800/1200 computer. This issue, **SoftSide Selections** for the Atari features:

- **Draw 7+** — This program lets you draw screens in graphics mode 7. A "blow-up" feature helps you draw details with ease.

- **Sharkey** — Gobble up fish in this undersea arcade-style game, but watch out for the electric eels!

- **Atari DV Bonus Program: Pix**
  Manipulate pre-drawn or original figures from a library — arrange them, change their colors and enlarge them — to create more intricate designs with this application of Peter J. Favaro's Nested Interpreter (Issue #43).

- **The SoftSide Adventure Series — Jack The Ripper II,** by Peter Kirsch
  Your grandfather chased the evil Ripper to a watery death in the River Thames, or so he thought. Now it seems that the Ripper lives again...

SOFTSIDE PUBLICATIONS, INC.
Amherst, New Hampshire